```cpp
#include "ANString.h"

int ANString::currentCount = 0;
int ANString::createdCount = 0;

ANString::ANString() {
    cap = 20;
    end = 0;
    str = new char[cap];
    str[end] = '\0';
    currentCount++;
    createdCount++;
}
ANString::ANString(const ANString& mstr) // Copy Constructor
{

        cap = mstr.cap;
        end = mstr.end;
        str = new char[cap];

        // Copy data over
        for (int i = 0; i < this->end; i++)
        {
            this->str[i] = mstr.str[i];
        }
        str[end] = '\0';
        currentCount++;
        createdCount++;
}
ANString::ANString(const char* cstr) {
    for (end = 0; cstr[end] != '\0'; ++end);
    cap = 20;
    if (end >= cap)
    {
        cap = 20 * (end / 20 + 1);
    }
    str = new char[cap];

    for (int i = 0; i <= end; ++i) {
        str[i] = cstr[i];
    }
    currentCount++;
    createdCount++;
}
ANString::~ANString() {

    delete[] this->str;
    currentCount--;
}
```

```
50
51  int ANString::length() {
52      return end;
53  }
54
55  int ANString::capacity() {
56      return cap;
57  }
58
59  char ANString::operator[](int index) {
60      if (index >= 0 && index < end) {
61          return str[index];
62      }
63      else {
64          return '\0';
65      }
66  }
67
68
69
70  ostream& operator<<(ostream& ostrm, const ANString& str) {
71      ostrm << str.c_str();
72      return ostrm;
73  }
74
75  bool ANString::operator<(const ANString& argStr) {
76      // TODO: you need to write.
77
78      int shortestCap = argStr.cap;
79
80      if (cap < argStr.cap)
81      {
82          shortestCap = cap;
83      }
84
85      for (int i = 0; i < shortestCap; i++)
86      {
87          if (str[i] > argStr.str[i])
88          {
89              return false;
90          }
91          else if (str[i] < argStr.str[i])
92          {
93              return true;
94
95          }
96
97      }
98
```

```
 99  }
100
101
102  bool ANString::operator>(const ANString& argStr) {
103      //TODO: you need to write.
104      int shortestCap = argStr.end;
105
106      if (end < argStr.end)
107      {
108          shortestCap = end;
109      }
110
111      for (int i = 0; i < shortestCap; i++)
112      {
113          if (str[i] < argStr.str[i])
114          {
115              return false;
116          }
117          else if (str[i] > argStr.str[i])
118          {
119              return true;
120
121          }
122
123      }
124
125      // IF WE GET HERE, ITS THE SAME SO FAR.
126      if (this->end > argStr.end)
127          return true;
128      return false;
129  }
130
131
132  bool ANString::operator==(const ANString& argStr) {
133
134
135      int shortestCap = argStr.cap;
136
137      if (cap < argStr.cap)
138      {
139          shortestCap = cap;
140      }
141
142      for (int i = 0; i < shortestCap; i++)
143      {
144          if (str[i] != argStr.str[i])
145          {
146              return false;
147          }
```

```
148         else if (str[i] == argStr.str[i])
149         {
150             return true;
151         }
152 
153     }
154 
155 }
156 
157 
158 ANString ANString::operator=(const ANString& rvalue) {
159     end = rvalue.end;
160 
161     if (end >= cap)
162     {
163         cap = 20 * (end / 20 + 1);
164         delete[] str;
165         str = new char[cap];
166     }
167 
168     for (int i = 0; i <= end; ++i) {
169         str[i] = rvalue.str[i];
170     }
171     return (*this);
172 }
173 ANString ANString::operator+(const ANString& rvalue) {
174     ANString Sum;
175     Sum.end = (end + rvalue.end);
176 
177     if (Sum.end >= Sum.cap)
178     {
179         Sum.cap = 20 * (Sum.end / 20 + 1);
180         delete[] Sum.str;
181         Sum.str = new char[Sum.cap];
182     }
183     int i = 0;
184     for (; i < end; i++)
185     {
186         Sum.str[i] = str[i];
187     }
188     for (int j = 0; j <= rvalue.end; j++)
189     {
190         Sum.str[i] = rvalue.str[j];
191         i++;
192     }
193 
194 
195     return Sum;
196 }
```

```
197
198  int ANString::getCurrentCount() {
199      return currentCount;
200  }
201
202  int ANString::getCreatedCount() {
203      return createdCount;
204  }
205
206  const char* ANString::c_str() const {
207      return str;
208  }
209
```